PROCEEDINGS OF SCIENCE

AuroraScience

Luigi Scorzato*
European Center for Theoretical Studies in Nuclear Physics and Related Areas (ECT)*
strada delle tabarelle, 286 - 38123 - Villazzano (Trento), Italy.
E-mail: scorzato@ect.it
(for the AuroraScience Collaboration)

AuroraScience is a research project at the crossroads of Computational Sciences and Computer Architecture. In this paper we introduce the AuroraScience project and report its status, with special consideration of its Lattice QCD applications.

The XXVIII International Symposium on Lattice Field Theory, Lattice2010
June 14-19, 2010
Villasimius, Italy

*Speaker.

PoS(Lattice 2010)039

1. Motivations

Over the years, the Lattice QCD community has accumulated an exceptional background both in the efficient use of the available computing resources, and also in the development of application driven HPC systems. Famous examples are the APE family of computers [1], QCDOC [2] and the recent QPACE project [3]. The experience accumulated in LQCD was beneficial also to other fields, as was shown by the development of the Janus architecture [4].

These projects not only provided efficient computing tools to the LQCD community, but also strengthened its know-how in the efficient use of computing resources, in a way that can only come from a close cooperation with the HW developers.

In recent years, the number of scientific fields, where HPC has become essential for competition, has increased enormously. As a consequence, investments in HPC are highly strategic to boost progress in many scientific disciplines at the same time. This is especially true if investments in hardware are combined with the building of a network that is able to efficiently disseminate HPC related know-how. In this context, those fields with a long HPC experience — such as LQCD — may be very beneficial for a larger community.

AuroraScience is a research project at the crossroads of Computational Sciences and Computer Architecture. It is a research and development project conducted by research groups from INFN, ECT*/FBK, University of Trento, University of Padova, E.Mach Foundation, the Agenzia Provinciale per la Protonterapia, who work in collaboration with the industrial partners Eurotech and Intel.

The collaboration builds on the combined know-how collectively available to the members of the collaboration on:

- design, development and operation of an application-driven high-performance computer system (e.g., the series of APE machines, developed by INFN together with DESY and the Paris Sud U.),

- algorithm development and physics analysis in computational areas of physics (Lattice Gauge Theory, Computational Fluid Dynamics, Molecular Dynamics), Quantitative Biology (Protein Folding), Bio-Informatics (Gene Sequencing) and Medical Physics.

AuroraScience can be seen as a scientific project enabled by leading-edge computational systems and by specific competences in the useful operation of these systems. The main goals of the project are the following:

- Tailor the architecture of a massively parallel computer system to the specific needs of a large class of regular computational problems. This is done by assembling a large number of latest generation multicore CPUs (Intel Westmere) and interconnecting them with a low-latency 3-D toroidal grid. The project uses the hardware recently introduced by Eurotech in their Aurora class of machines. The AuroraScience toroidal network environment (ATNW) [5] is built on top of the TNW-project, which has been used in QPACE and then within AuroraScience. The network processor is encoded in a firmware that has been installed on Aurora FPGA.

- Procure and operate a medium-size Aurora prototype machine, optimized for AuroraScience. The envisaged installation will have in excess of 100 processing nodes with a target peak performance of some tens of Tflops. The project considers the prototype as a convincing test proof that large-scale systems can be efficiently operated by a combination of off-shelf high-end processors and application-driven interconnects. Following successful operation of the prototype, the project will consider enlarged collaboration options, with the main goal of developing a Pflops class machine optimized for scientific applications and fully available to basic science.

- Modify, tune, port and optimize for the AuroraScience prototype key computational algorithms in the scientific areas described above, and show effectiveness for state-of-the-art scientific simulations.

2. Development Of The Computing System

2.1 The Aurora Architecture

Aurora is a multilevel, high performance parallel architecture that has been designed to deliver very high sustained performance both for general purpose parallel applications and for several high end massively parallel programs, such as those used for Lattice-QCD (LQCD) simulations or fluid-dynamics modeling for which efficient scaling to thousands of processing nodes is mandatory.

The Aurora architecture is based on processing elements (PE) powered by latest generation multi-core processors harnessed by a combination of several interconnection networks: a general purpose high end interconnection system (Infiniband), a 3D grid network. Moreover, a fast synchronization system is under development. These cooperate to meet the conflicting requirements of flexibility for traditional applications and good scaling properties for massively parallel programs. This section briefly describes Aurora at system level and presents some details of the PEs. The interconnection structure is described in details in [5].

The Aurora processing element uses the latest Intel multi-core Westmere processors [6]. The CPU/RAM block consists of two 6 cores Intel Westmere chips, each coupled to three banks of DDR3 memory, for a total of 12GB. Each processor hosts 12MB of L3 cache and each core has 256KB of L2 cache. The processor clock is 3.33 GHz. The Intel Tylersburg Southbridge is the central hub of the Aurora node, handling all data transfers among the main functional blocks: CPU/RAM, high performance network interface, general purpose network, additional peripherals. A total of three QPI links connect the CPUs together and each CPU with the Tylersburg. The interface for the 3D network and for the system-wide synchronization system connects to the PE over one x8 PCI Express Gen 2 link (with a bandwidth of about 8 GByte/sec) made available by the Tylersburg bridge.

In Aurora, PEs are hosted in a 6U rack-mountable enclosure (one chassis). Each chassis has a symmetrical design and contains two sets of 16 node-cards, each connected by a backplane and one root card. The root card integrates a Mellanox Infiniband switch with 36 ports; 16 ports are internally connected to the node-cards, while the remaining 20 are made available for external connections. The backplane is used to provide the physical connection (data signals, power) between

all the elements; on the edges of the backplane, a set of high density connectors is used to deliver the torus network signals via high speed flat cables.

The power dissipated by each chassis (32 boards) is of the order of $\simeq$ 10KW. Heat is removed by direct liquid cooling, which means that each node-card is in contact with a 'coldplate', through which water flows. Very high heat transfer efficiency is possible thanks to the very favorable heat capacity of water. Direct liquid cooling introduces additional complexity but also significant benefits, mainly in terms of reduced total energy budget and reliability. In fact, the water flux per each node-card is about 80 l/h and typical in/out temperatures are 25/30 °C. This means that free cooling is possible for most part of the year and in most climates.

A fully populated 48U rack holds up to eight chassis and support equipment, corresponding to a peak performance of approximately 40 Tflops.

2.2 The Trento Installation

The Aurora installation in Trento is located in the premises of the new Interdisciplinary Laboratory for the Computational Sciences (LISC), which is operated jointly by the FBK and the University of Trento.

A dedicated circuit to treat and cool the water has been installed by the FBK under the direction of TESI engineering. The system implements the techniques of free cooling and heat recovery, in order to further save energy. As of today, two chassis are installed corresponding to a peak performance of 10 Tflops. Four servers are connected to the rack via Infiniband and are dedicated to operate a parallel gpfs file system. A dedicated service ("atnadmin") has been designed to manage the partitioning of the system with respect to the torus network connectivity: the user can list available and active partitions, monitor status in terms of active processes, activate or delete partitions. A customized queue system has been developed to manage jobs submission.

3. Scientific Applications

The scientific applications studied within AuroraScience are Lattice QCD (ECT* and INFN groups), Biophysics and Molecular Dynamics (UNITN), Lattice Boltzmann Methods (INFN), Quantum Monte Carlo simulations of Many-Body Nuclear Physics (UNITN), Lorentz Integral Transform methods for the study of Few-Body Nuclear Physics (UNITN), Bioinformatics (FEM) and Monte Carlo simulations for Radiotherapy (ATreP).

A challenge of this project is to exploit the knowledge of the hardware to boost performance of the above scientific applications. However, modern scientific codes typically include a large variety of features, and put considerable emphasis on ensuring maintainability and portability. It is very important that also highly optimized, hardware specific, solutions respect these constraints.

In this paper we concentrate in one specific case: that of porting tmLQCD – a widely used Lattice QCD software package [7] – to exploit the Aurora architecture and in particular the ATNW [5] communication network. In this case there are two distinct issues. First, the LQCD algorithms should be adapted to exploit the architectural advantages of Aurora. These algorithmic ideas will be useful also for other systems and should be implemented in the tmLQCD package in a portable way. This is discussed in Section 3.1. A second issue is the one of adapting the tmLQCD code to use the low level ATNW communication routines. Much of this work is useful for other applications

besides LQCD. It is therefore a good idea to collect the corresponding tools in a library, which is described in Section 3.2.

3.1 Alternative Parallelization Schemes

The hardware features of a parallel machine affect strongly the optimal parallelization strategy of the Dirac operator. In a system with a high ratio $\rho =$ latency of the communications / time for floating point operations, it is convenient to collect all data to be sent in big chunks. This is typical of PC clusters, where all the borders are exchanged after a full sweep over the space-time volume. The price of latency is payed only once, but cannot be hidden.

The opposite case is represented by machines in the APE family, where the ratio ρ is very low. Here one can afford exchanging the data just before they are needed. This allows an optimal overlap of communications and computations, which hides the communication latency.

Modern, strongly coupled and massively parallel computing systems, like QPACE and AURORA, have intermediate values of ρ. And the optimal parallelization must be different from both the previous ones. One possibility is to split the computation into 3D slices. If we assume that *t* (=time) if the outmost loop, the computation in one 3D times-lice can be partially overlapped with the exchange of the *xt*, *yt* and *zt* borders. This strategy is appropriate, because the time to compute the timelike links in the smallest possible local times-lice (4^3) is about $0.5\mu s$, and for more realistic local lattices, it becomes quickly much larger that the latency of the network ($\sim 1\mu s$). This method was first proposed in [8], and we have implemented it in the tmLQCD code using standard MPI tools.

A second improvement was the multi-thread parallelization of the critical volume loops (via openMP). The possible advantages come from saving both time and memory for border exchanges and from the higher flexibility of the parallelization (the number of cores does not need to divide exactly the number of points in the edges). Disadvantages may be expected because of the higher amount of serial parts, overheads and memory access conflicts. Also this feature was implemented in the tmLQCD code.

Some results of the corresponding benchmark tests are given below.

3.2 The Torus Library

The ATNW communication routines, described in [5], can be accessed directly by any high level programming language. However, in order to minimize the communication overhead, they are transparent to many hardware features. In particular they support a limited buffer, they can only access nearest neighbors boards, and require different programming strategies for different problem sizes.

Most scientific software uses MPI in an oblivious way with respect to those characteristics. Porting the ATNW routines efficiently to these codes, without breaking important functionalities and maintainability is a challenge. To this end we developed a special library (toruslib) that provides a convenient interface between the ATNW routines and a typical MPI application. This library does not aim at a general support of MPI. Instead it implements via the ATNW only those regular and nearest-neighbors communications that can be ported efficiently. The other communications are left to the Infiniband network.

Some functions in this library respect exactly the same prototypes as the corresponding MPI functions and can be accessed with a simple recompilation of the program. Other functions, which are meant to enable more flexibility in the optimizations, are provided to be used directly in the application code.

3.3 Results On Aurora

In this section we report mainly on tests done on the benchmark code for the (even-odd preconditioned) Hopping Matrix. We briefly mention a real application towards the end.

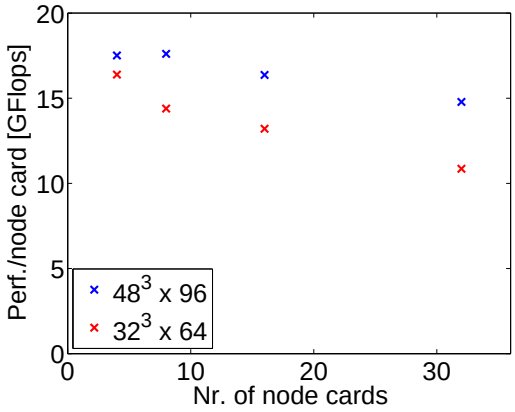
The deep hierarchical structure of modern high-end processors – and in particular the Westmere – opens the possibility to many different parallelization strategies inside the node-card. For example, 12 different MPI processes can be assigned to the 12 processing cores of a single node-card. Moreover, these MPI processes can be organized in any topology. Alternatively, all the 12 cores can be assigned to the same MPI process, but to different threads (implemented via openMP, as mentioned in Section 3.1). Moreover, many mixtures of the above approaches is possible. We performed an extensive series of tests on the benchmark code and found the following conclusions. If possible, the most efficient parallelization is given by 12 MPI processes, organized in a 2x2x3 grid. If this is not possible (e.g. on a $32^3 \times 64$ lattice), the best option is usually one of the following (in the notation M-nr. of MPI processes, C-nr. of threads per MPI process): $M8C1$, $M4C2$, $M4C3$, $M2C5$. The combination $M2C6$ is rarely efficient, and the combinations $M3C4$ and $M6C2$ always performed very badly in all our tests.

Currently, the most efficient algorithm for the Hopping Matrix multiplication , which is used for production runs in the Blue Gene/P systems, is called half-spinor in [7]. We compared it to the newly developed time-split algorithm described in Section 3.1. The performance gain on the largest physical lattice that we considered ($48^3 \times 96$) is very satisfactory: a factor between 1.8 (using 4 node cards) and 1.5 (using one chassis = 32 node-cards). On smaller physical lattices ($32^3 \times 64$) the gain goes down to a factor between 1.2 and 1.6. On the smallest physical lattices ($16^3 \times 32$) the half-spinior algorithm is more efficient. This behaviour is expected, since the time-split algorithm is optimized for cache usage.

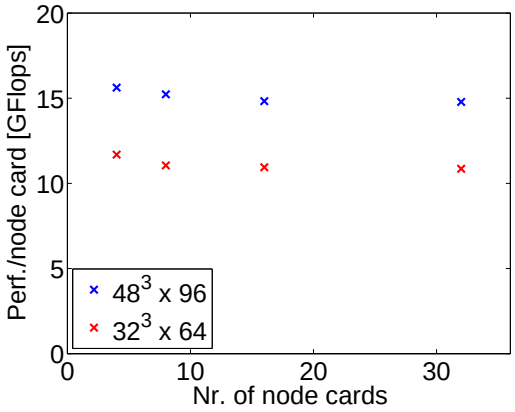
With the new algorithm we obtained up to 12% of the peak performance (which is 150 Gflops / node-card). In Figure 1(a) we show an example of strong scaling, in which the problem size is kept fixed and distributed across an increasing number of nodes. In Figure 1(b) we show an example of weak scaling, in which the size of the problem in a single node is kept fixed while the full problem size grows proportionally to the number of nodes. Weak scaling is nearly ideal, which means that the Infiniband network is still very efficient up to the size that we considered. Strong scaling is sensitive to more effects, but it appears to be still quite good up to 32 nodes.

The time-split algorithm is now being used in Aurora to produce dynamical $N_f = 4$, $L = 24$, Twisted Mass, Iwasaki gauge configurations with the goal of computing non perturbatively the renormalization factors. With these parameters the code is performing a factor 1.5 better than the version using the half-spinor algorithm.

We also tested the code that uses the 3D ATNW network, via the torus libraries. The ATNW network is currently under development within the AuroraScience project and the full bandwidth presently available (see [5]) is still too low to be competitive with the Infiniband switches on the small systems that we are considering. However, the good news is that the effective bandwidth



(a) Example of strong scaling, as explained in the main text. Errors are on the range of 5%.



(b) Example of weak scaling. The labels refer to the size of the problem on 32 nodes. Errors are expected on the range of 5%.

(which includes all overheads and which is measured in the benchmark code) is not much below the full available bandwidth.

Acknowledgments

The AuroraScience project is funded by the Provincia Autonoma di Trento (PAT) and the Istituto Nazionale di Fisica Nucleare (INFN), in the framework of an agreement with the Fondazione Bruno Kessler (FBK). See `web.infn.it/aurorascience`. LS is a member of the Interdisciplinary Laboratory for Computational Sciences (LISC). We wish to thank our collaborators at Eurotech and Intel. We are grateful to the members of QPACE for many valuable discussions and design ideas, and we thank PetaQCD and QPACE for stimulating meetings.

References

[1] F. Belletti et al., *Computing for LQCD: apeNEXT*, Computing in Science & Engineering **8** (2006) 18.

[2] P.A. Boyle et al., *Overview of the QCDSP and QCDOC computers*, IBM J. Res. & Dev. **49** (2005) 351.

[3] H. Baier et al., *QPACE – a QCD parallel computer based on Cell processors*, `arXiv:0911.2174 [hep-lat]`.

[4] F. Belletti et al., *JANUS: an FPGA-based System for High Performance Scientific Computing*, Computing in Science & Engineering **11** (2009) 48.

[5] M. Pivanti, F.S. Schifano, H. Simma, *The AuroraScience Project: The Machine*, `PoS(LAT2010)038`.

[6] http://processorfinder.intel.com

[7] K. Jansen and C. Urbach, *tmLQCD: a program suite to simulate Wilson Twisted mass Lattice QCD*, Comput. Phys. Commun. **180** (2009) 2717. CPHCB,180,2717;

[8] F. Belletti et al., *QCD on the Cell Broadband Engine*, `PoS(LAT2007)039`. arXiv:0710.2442 [hep-lat].